BAB III
ANALISA DAN PERANCANGAN SISTEM

Pada bab ini akan menjelaskan analisa dan perancangan skema yang diperlukan untuk tahapan dalam pengembangan mekanisme akses dan manajemen *E-learning* berbasis web.

3.1 Analisis Masalah

Penelitian ini bertujuan untuk mengembangkan mekanisme akses pada *E-learning* berbasis Docker *container* yang semula menggunakan *port* sehingga menjadi sub domain dengan menggunakan *reverse proxy*. Penelitian ini juga bertujuan untuk melakukan pengembangan mekanisme manajemen yang semula menggunakan *script* Bash menjadi berbasis web. Sehingga pengguna dan administrator dapat lebih mudah untuk mengakses dan melakukan manajemen pada *E-learning*.

3.2 Analisis Sistem

3.2.1 Mekanisme Akses

Penggunaan *Domain Name System* (DNS) membuat alamat suatu website menjadi lebih mudah diakses. Alamat IP sebagai alamat pada tiap-tiap *host* dalam komunikasi data menyulitkan pengguna untuk mengingat banyak nomor IP untuk mengakses suatu *host* dalam jaringan intranet ataupun internet. DNS merupakan jenis sitem yang melayani pemetaan IP a*ddress* ke FQDN (*Fully Qualified Domain Name*) dan sebaliknya. Sebagai contoh sebuah *host* memiliki alamat IP 172.18.0.1 dan FQDN "www.moodlenet.com". Nama "www.moodlenet.com" tentu akan lebih mudah diingat dari pada alamat IP nya.

Container pada Docker merupakan suatu ruang virtual yang menjalankan proses terisolasi yang dapat berjalan di atas berbagai *platform* seperti Windows, Linux, dan UNIX. Penelitian terdahulu menghasilkan bahwa Docker dapat mereduksi penggunaan sumber daya CPU dan memori dibanding menggunakan virtualisasi berjenis Hypervisor [3]. Berbeda dengan virtualisasi hypervisor, container pada Docker dapat berkomunikasi dengan jaringan public dengan metode *port publishing*. Metode ini adalah melakukan pemetaan *port* pada *host* pada container. Penggunaan *port* sebagai memetakkan *E-learning* menjadikan

pengaksesan dan proses administrasi lebih sulit dikarenakan harus menghafal *port* pada tiap-tiap *E-learning*.

Reverse proxy adalah stasiun transfer informasi jaringan yang menjembatani antara klien dan *server* untuk saling bertukar pesan [27]. Salah satu perangkat lunak yang digunakan seagai *reverse proxy* adalah Nginx. Selain sebagai web *server*, Nginx juga dapat melakukan peran sebagai *reverse proxy*. Dengan menggunakan Nginx penggunaan *port* sebagai jalur pengaksesan *E-learning* dapat dirubah yang semula melalui *port* menjadi subdomain.

3.2.2 Mekanisme Manajemen

Untuk melakukan administrasi dan manajemen *server* dapat dilakukan secara *on-site* maupun *remote*. SSH adalah salah satu protokol yang dapat digunakan untuk melakukan *remote* suatu *server*. Dalam penelitian sebelumnya, untuk melakukan manajemen *E-learning* terdapat program berbasis *script* Bash yang berarti harus melakukan *remote* melalui SSH untuk menjalankan program tersebut ketika tidak berada pada lingkungan *server*. Hal tersebut menjadikan proses administrasi kurang fleksibel. Hal tersebut tentu akan berbeda ketika manajemen container dapat dilakukan dengan aplikasi berbasis web yang menjadikan proses manajemen lebih fleksibel.

3.2.3 Analisa Kebutuhan Fungsional

1. Sistem Operasi

Sistem operasi yang digunakan untuk melakukan implementasi sistem adalah Ubuntu *Server* 16.04 LTS yang merupakan salah satu distro berbasis Debian. Arsitektur Ubuntu *Server* yang digunakan adalah x64. Dipilihnya arsitektur x64 karena virtualisasi Docker *Container* hanya dapat berjalan pada arsitektur tersebut.

2. DNS Server

DNS *Server* yang digunakan pada implementasi sistem pada penelitian ini adalah bind9. Bind9 adalah perangkat lunak dengan sumber terbuka yang berguna untuk mempulikasikan informasi DNS pada internet serta dapat berjalan di berbagai *platform* yaitu Windows, Linux, dan UNIX.

3. Web Server

Web *server* dibagi menjadi dua pada penelitian ini, yaitu Apache dan Nginx. Web *server* Apache pada penelitian ini digunakan sebagai web *server E-learning* yang berjalan didalam *container* yang sama dengan Moodle. Berbeda dengan Nginx, pada penelitian ini perangkat lunak ini memiliki dua peran, yaitu sebagai web *server* dan *reverse proxy*. Nginx berperan sebagai web *server* untuk melayani dan menjalankan aplikasi manajemen *E-learning*. Peran yang kedua yaitu Nginx sebagai *reverse proxy* yang berfungsi untuk meneruskan dan menejemahkan *request* pengguna yang ditujukan untuk Moodle.

4. Basis Data

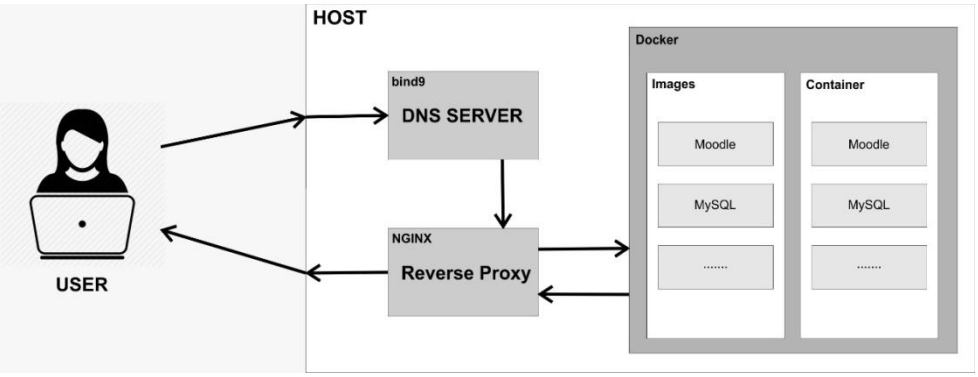
Basis data yang digunakan pada penelitian ini adalah MySQL. MySQL dibagi menjadi 2 yaitu MySQL yang digunakan sebagai penyimpanan data Moodle yang berjalan pada *container* yang terpisah dengan Moodle. Yang kedua yaitu MySQL yang digunakan sebagai menyimpan data Aplikasi Manajemen *E-learning*.

5. Bahasa Pemrograman

Bahasa pemrograman yang digunakan untuk mengembangkan aplikasi manajemen *E-learning* adalah Python dengan versi 3.0. Python adalah salah satu bahasa pemrograman interpreter dan berorientasi objek yang berkembang secara luas dan populer di *science* dan *engineering*. Python memiliki sintak dan tipe data yang komplek. Selain itu Python juga dibekali dengan *package* dan *library* yang cukup lengkap [28]. Pada penelitian ini ada beberapa *package* dan *library* yang digunakan untuk menyelesaikan sistem yang dibangun yaitu library Docker-py (Docker SDK *for* Pyhton). Docker-py adalah suatu *library* pyhton untuk Docker *Engine* API. *Library* ini memungkinkan pyhton untuk melakukan perintah yang ada pada Docker.

3.3 Rancangan Sistem

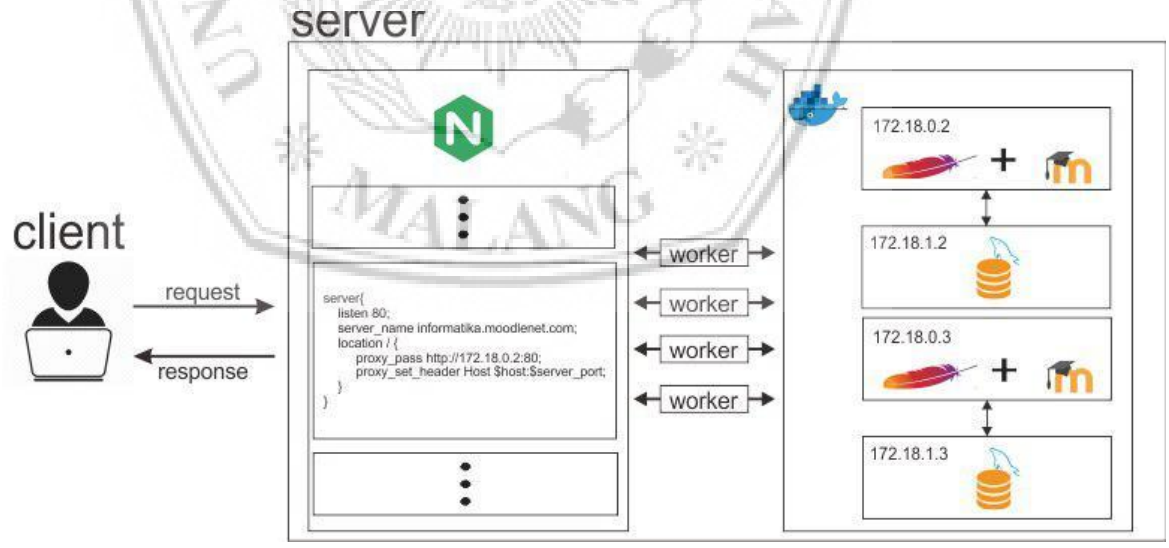
Rancangan sistem pengembangan mekanisme akses dan manajemen *E-learning* berbasis linux *container* dibagi menjadi 2 sub sistem, yaitu sistem mekanisme akses *E-learning*, dan sistem manajemen *E-learning*. Gambar 4 menunjukkan keseluruhan rancangan sistem yang dibangun. Berikut merupakan penjelasan dua sub sistem yang dikembangkan.



Gambar 1. Rancangan sistem

3.3.1 Rancangan *Reverse proxy*

Mekanisme akses *E-learning* berbasis DNS memanfaatkan kemampuan Nginx sebagai *reverse proxy*. Cara kerja Nginx sebagai *reverse proxy* adalah meneruskan *request* klien pada *server* yang sesuai. Pada penelitian ini yang dimaksud *server* yang sesuai adalah masing-masing *E-learning* yang berjalan pada tiap-tiap *container*. Proses *request* yang dilakukan oleh *client* berupa URL akan diterima oleh Nginx dan akan diteruskan pada *server* yang dimaksud. Misal client melakukan *request* URL "informatika.moodlenet.com", URL tersebut akan diterima oleh Nginx dan akan diterjemahkan dan diteruskan pada IP dan *port* yang terdaftar pada konfigurasi *sites* Nginx.



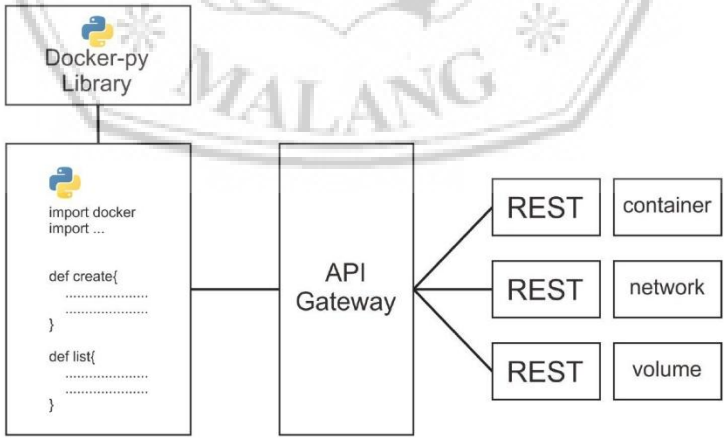
Gambar 2. Mekanisme Nginx sebagai *reverse proxy*

Alur kerja *reverse proxy* diilustrasikan seperti pada gambar 5. Pertama kali adalah *client* melakukan *request* dengan mengirimkan URL *E-learning* yang akan diterima oleh Nginx. URL yang diterima oleh Nginx akan dicocokkan pada

konfigurasi *sites*. Jika ditemukan kecocokan antara URL dan konfigurasi, maka Nginx menciptakan suatu proses yang dinamakan *worker* yang berfungsi untuk melayani permintan client. *Worker* yang berfungsi sebagai jembatan antara jaringan luar dan internal melakukan request kepada alamat IP yang telah diterjemahkan oleh Nginx yaitu kepada web *server* apache yang berjalan di dalam kontainer. Web *server* apache bertugas untuk menyajikan moodle dan data yang tersimpan pada MySQL yang berada pada kontainer lain dan dikirimkan kepada *worker* yang melakukan *request*.. Setelah mendapatkan respon dari apache, *worker* meneneruskan respon tersebut kepada *client*. Kelebihan dari proses tersebut menjadikan IP internal dari Docker tidak diketahui oleh *client* karena alamat IP yang diterima oleh *client* adalah alamat IP *server*.

3.3.2 Rancangan Aplikasi Web

Docker Python SDK adalah *library* Python yang digunakan untuk berkomunikasi dengan Docker *Daemon* melalui Docker *Engine* API yang membolehkan program untuk mengirim perintah apapun pada Docker. Docker *engine* API adalah suatu API yang disediakan oleh Docker agar program yang memanfaatkan library Docker SDK Python dapat berkomunikasi dengan Docker *daemon*[9]. Docker *Engine* API berfungsi sebagai protokol penerjemah menjadi REST yang akan melakukan perintah pada Dcoker *daemon*. Hal tersebut diilustrasikan pada gambar 6.



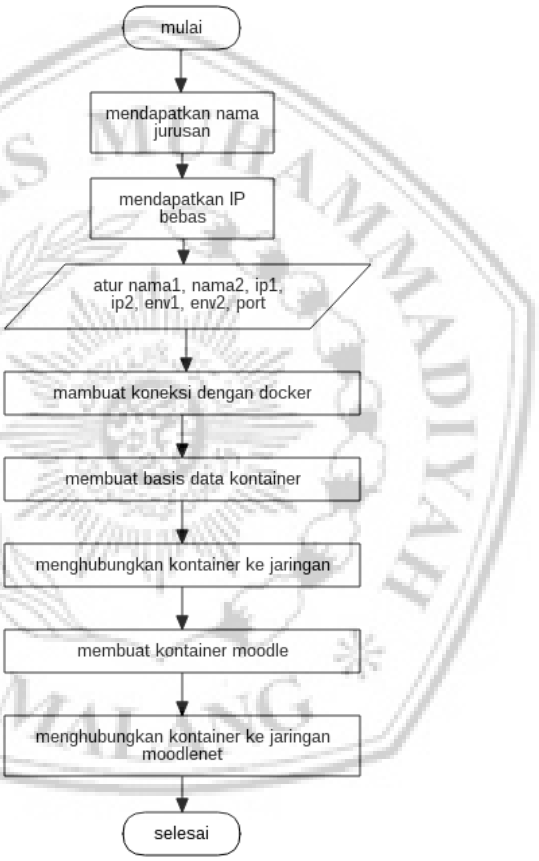
Gambar 3. Alur kerja Docker Python SDK

Pada gambar 6 menjelaskan tentang alur kerja Docker Python SDK yang menyediakan abstraksi perintah yang cukup lengkap untuk melakukan manajemen dan administrasi Docker. Untuk dapat menggunakan *library* ini, diperlukan

instalasi *library* Python SDK terlebih dahulu. Agar dapat terhubung dengan API, diperlukan mendeklarasikan variabel dengan memanggil modul DockerClient. Beberapa modul yang digunakan dalam implementasi aplikasi manajemen *E-learning* adalah:

1. *Create Container*

Modul ini berfungsi untuk membuat kontainer yang menjalankan *E-learning*. Dalam modul ini program akan membuat dua buah kontainer yang terdiri dari web *server* dan *database*.



Gambar 4. Diagram alir modul *create container*

Gambar 7 adalah diagram alir yang merepresentasikan proses pembuatan kontainer *E-learning*. Proses di awalai dengan mendapatkan nilai nama *container* yang didapat dari inputan *user* pada web manajemen *E-learning*. Setelah itu proses dilanjutkan dengan mendapatkan IP yang sedang tidak terpakai oleh kontainer lain. Nilai IP pada proses ini didapatkan dari modul ip_alocator. Lalu proses selanjutnya adalah menentukan nilai paramter yang diperlukan untuk

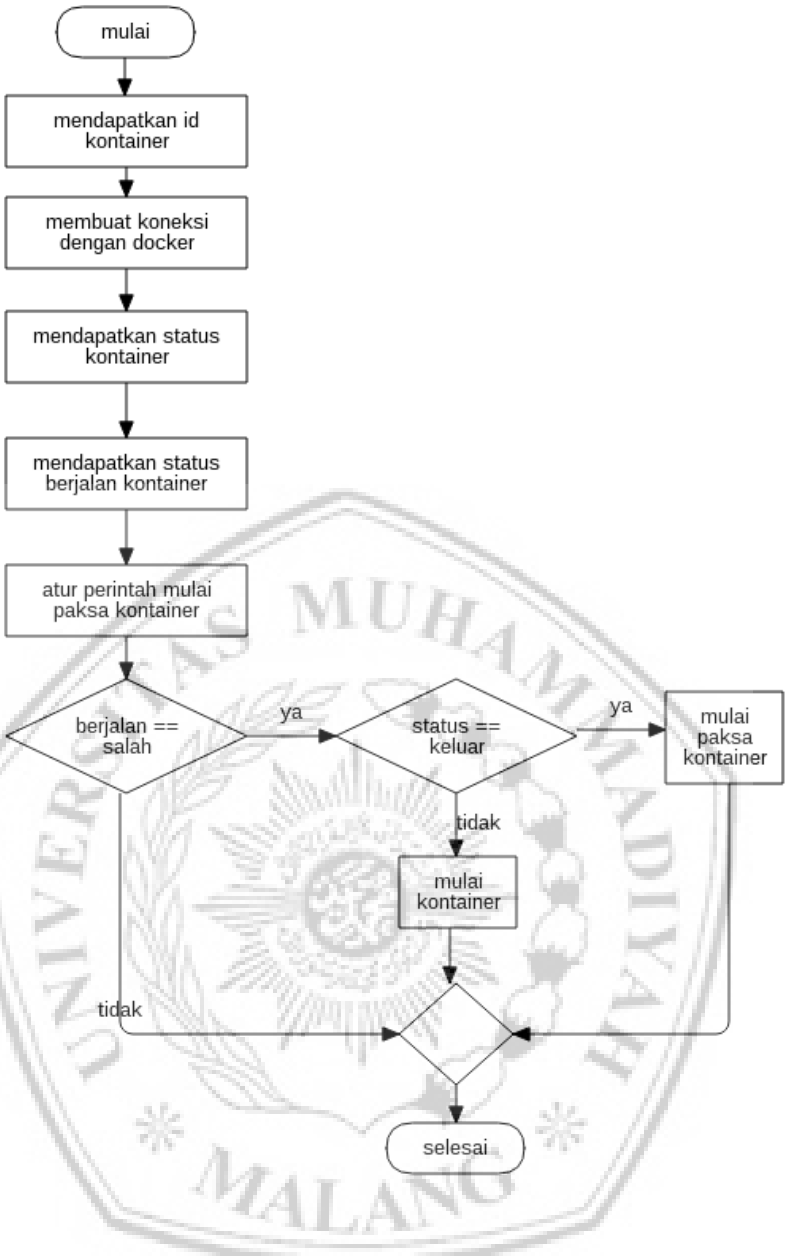
membuat kontainer, yaitu nama, IP, *port*, dan *environment*. Setelah parameter terpenuhi, proses selanjutnya adalah membuat koneksi dengan Docker API. Setelah terhubung, proses selanjutnya adalah membuat dua kontainer baru sesuai dengan parameter yang telah disiapkan. Setelah kontainer terbuat, program mengirimkan perintah kontainer untuk terhubung pada jaringan moodlenet. Setelah itu proses berakhir.

2. *Start Container*

Modul ini berfungsi untuk memulai kontainer yang sebelumnya sudah dibuat dan sedang dalam kondisi *exited*, atau mati.

Gambar 8 merupakan diagram alir yang menggambarkan proses *start container*. Proses dimulai dengan mendapatkan nilai ID *container* yang akan di *start*. Selanjutnya adalah mendapatkan *state* status dari pada *container* tersebut. Lalu mendapatkan *running* status *container* tersebut. Lalu dari nilai yang didapatkan pada dua proses sebelumnya akan ditentukan langkah selanjutnya dengan membandingkan nilai yang di dapat pada status *running*. Jika nilainya *False* maka dilanjutkan dengan membandingkan nilai status *state*. Jika nilainya *True* maka proses akan diakhiri. Dalam kondisi *False*, jika status *state* nilainya sama dengan "*exited*" maka proses dilanjutkan dengan melakukan *start* kontainer dan proses diakhiri. Jika nilai tidak sama dengan "*exited*" maka proses selanjutnya dilakukan dengan *force start container* dan proses diakhiri.

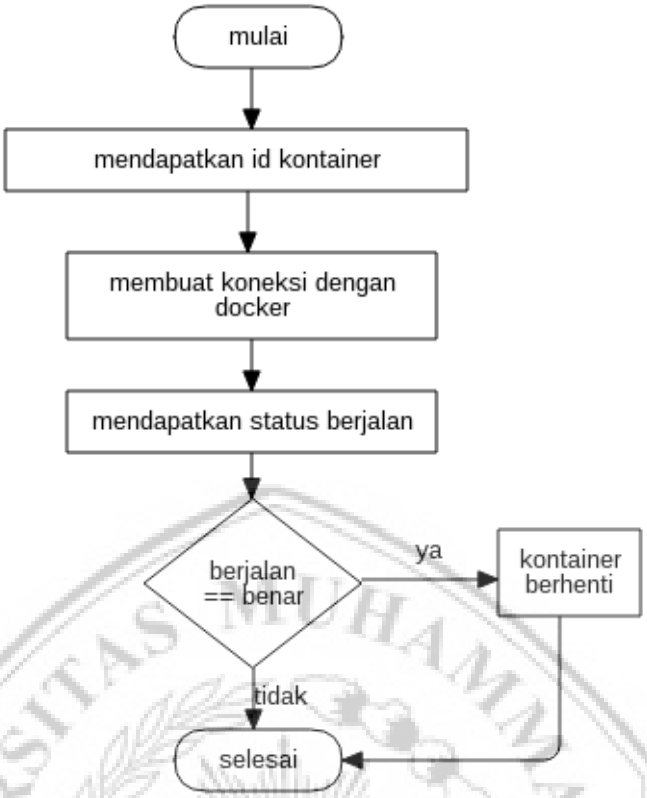
Perbedaan *start* dan *force start* pada proses ini adalah *start* dijalankan ketika kontainer berhenti secara normal dan sesuai prosedur. Sedangkan *force start* dijalankan ketika *container* tersebut berhenti tidak sesuai prosedur dan tidak bisa dijalankan dengan hanya perintah "Docker *start*". Hal tersebut dikarenakan proses berhenti kontainer sebelumnya dilakukan dengan cara paksa seperti proses Docker atau *server* langsung dimatikan. Hal tersebut menyebabkan proses Apache yang berada pada kontainer masih tersimpan pada apache.pid. Maka langkah yang ditempuh adalah mengirimkan perintah untuk memulai *container* dengan perintah *start* dan dilanjutkan secara langsung dengan menghapus berkas apache2.pid tanpa sela waktu sehingga *container* dapat berjalan secara normal kembali.



Gambar 5. Diagram alir start *container*

3. *Stop Container*

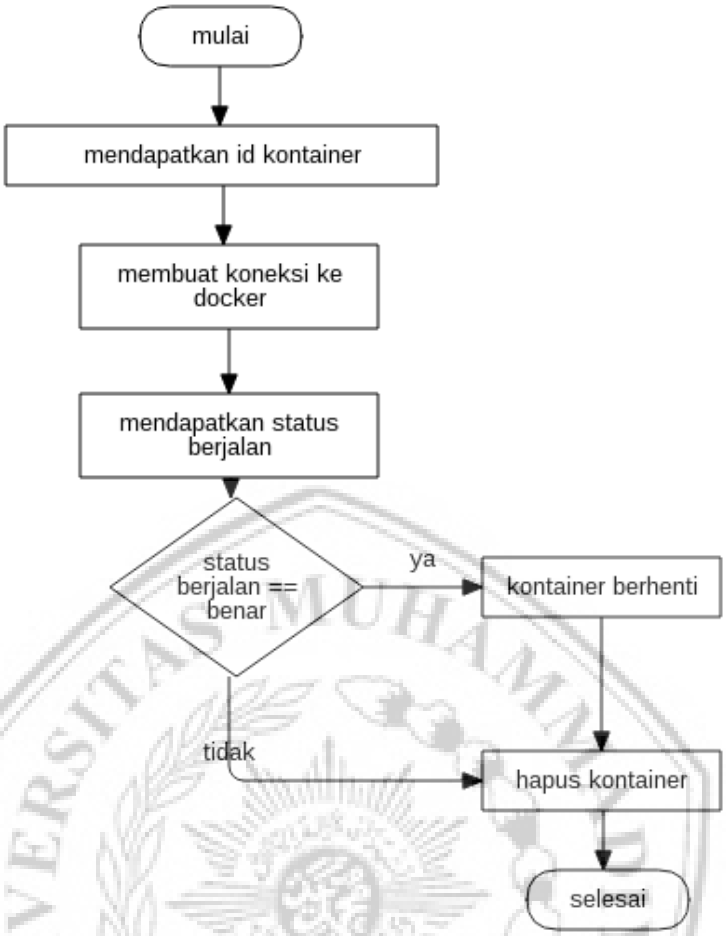
Pada gambar 9 menjeaskan tentang diagram alir *stop container*. Modul ini berfungsi untuk melakukan stop kontainer yang sedang berjalan. Proses dimulai dengan mendapat id *container*. Lalu membuat koneksi dengan Docker. Dilanjutkan dengan mendapatkan *running state* dari *container* tersebut. Jika *running state* sama dengan *True*, maka proses dilanjutkan dengan *stop container* lalu proses berhenti. Jika *running state* sama dengan *False* maka proses akan langsung dihentikan.



Gambar 6. Diagram alir stop *container*

4. *Delete Container*

Pada gambar 10 menjelaskan diagram alir modul *delete container*. Modul ini berfungsi untuk menghapus kontainer. Proses dimulai dengan mendapatkan *id container*. Lalu dilanjutkan dengan membuat koneksi dengan Docker. Lalu proses selanjutnya adalah medapatkan *running state* dari *container*. Jika *running state* sama dengan *True* maka prose dilanjutkan dengan *stop* kontainer lalu *delete* kontainer, dan proses diakhiri. Jika *running state* sama dengan *False* maka dilanjutkan ke proses *delete* kontainer lalu proses diakhiri.



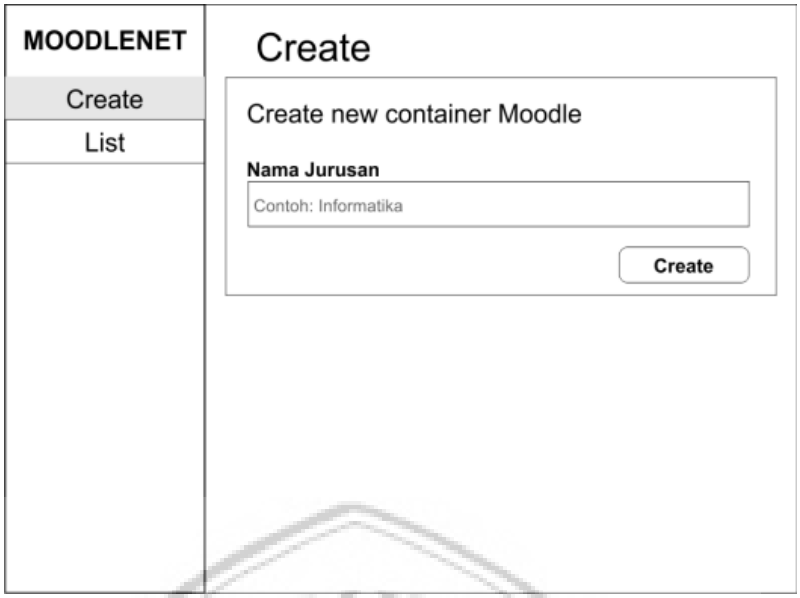
Gambar 7. Diagram alir delete *container*

3.4 Rancangan Antarmuka

Berikut merupakan rancangan antarmuka aplikasi Moodlenet yang digunakan untuk melakukan manajemen *E-learning*.

1. *Create*

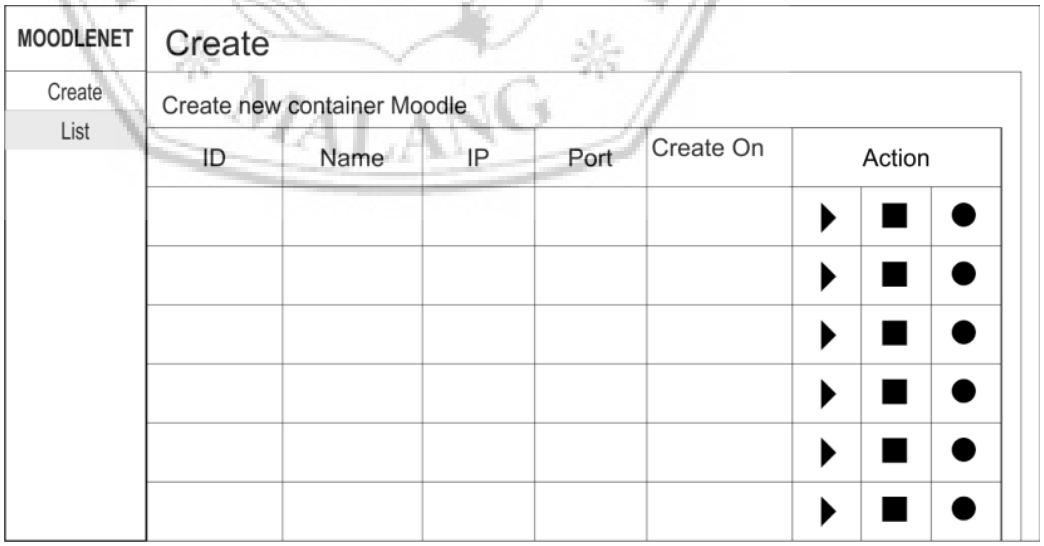
Halaman *create* memuat *form* yang digunakan untuk menentukan nama dari *container* Moodle. Halaman *create* ini akan menjadi halaman utama ketika *website* diakses. Gambar 11 merupakan rancangan antarmuka dari halaman *create* pada Moodlenet.



Gambar 8. Rancangan antar muka halaman *create*

2. *List*

Halaman *list* digunakan untuk menampilkan daftar *container* yang terdaftar pada sistem Moodlenet. Selain untuk menampilkan daftar *container*, disediakan tombol *action* yang terdiri dari tombol *start*, *stop*, dan hapus. Berikut merupakan rancangan antarmuka halaman *list* yang diilustrasikan pada gambar 12. Masing-masing tombol *action* yang diwakili dengan tombol warna hijau berfungsi untuk *start* kontainer, biru untuk *stop*, dan merah untuk menghapus kontainer.



Gambar 9. Rancangan antarmuka halaman list

3.5 Metode Pengujian Dan Analisis Hasil Uji Coba

Pengujian dilakukan untuk menganalisis performa sistem, *throughput* dan waktu respon serta fungsionalitas aplikasi manajemen *E-learning*.

3.5.1 Pengujian Performa Sistem

Tahapan ini dimaksudkan untuk mengevaluasi fungsionalitas, tingkat akurasi, dan performa dari sistem tersebut. Salah satu pengujian yang dilakukan adalah pengukuran sumber daya CPU dan memori terhadap penerapan sistem yang baru. Peneliti juga akan membandingkan performa dan kinerja *E-learning* yang menerapkan sistem baru dengan sistem yang terdahulu dengan parameter waktu respon dan *throughput.* Pengujian tersebut dilakukan dengan menggunakan aplikasi JMeter. Selain itu, pengujian kemampuan *server* dalam memuat *container* juga akan dilakukan untuk mengetahui batas maksimal kapasitas *server* dalam memuat *container*. Berikut adalah tabel penjabaran tahap pengujian.

Tabel 4 merupakan pengujian performa sistem dengan tujuan untuk mengukur penggunaan CPU dan Memori dengan skenario pertama yaitu terdapat satu *E-learning* yang berjalan dengan *user* pengakses sebanyak 100 *user*. Skenario kedua yaitu yaitu terdapat tiga *E-learning* yang berjalan dengan *user* pengakses sebanyak 100 *user* pada masing-masing *E-learning*. Dan skenario ketiga yaitu terdapat lima *E-learning* yang berjalan dengan *user* pengakses sebanyak 100 *user* pada masing-masing *E-learning*.

Tabel 1. Pengujian performa sistem

JUMLAH *CONTAINER*		**1**	**3**	**5**
Jumlah *User* per *Container*		100	100	100
CPU	IP			
	DNS			
MEMORI	IP			
	DNS			
Total *User*		100	300	500

Tabel 5 merupakan pengujian waktu respon dan *throughput* sistem dengan tujuan untuk mengukur *waktu respon* dan *throughput* dengan skenario pertama yaitu terdapat satu *E-learning* yang berjalan dengan *user* pengakses sebanyak 100 *user*. Skenario kedua yaitu terdapat tiga *E-learning* yang berjalan dengan *user* pengakses sebanyak 100 *user* pada masing-masing *E-learning*. Dan skenario

ketiga yaitu yaitu terdapat lima *E-learning* yang berjalan dengan *user* pengakses sebanyak 100 *user* pada masing-masing *E-learning*.

Tabel 2. Pengujian waktu respon dan *Throughput* sistem

JUMLAH *CONTAINER*		**1**	**3**	**5**
Jumlah *User* per *Container*		100	100	100
Waktu respon	IP			
	DNS			
Throughput	IP			
	DNS			
Total *User*		100	300	500

3.5.2 Pengujian *Black Box*

Pengujian *black box* merupakan pengujian yang dilakukan untuk menguji fungsionalitas *input* dan *output* pada perangkat lunak. Penguji mendefinisikan sekumpulan kondisi *input* lalu melakukan pengujian terhadap fungsi tersebut sehingga menghasilkan *output* yang nilainya dapat dievaluasi. Pengujian dilakukan dengan melakukan pengujian modul *create* kontainer, *start* kontainer, *stop* kontainer, dan *delete* kontainer. Pada tabel 6 akan dijelaskan rancangan daftar pengujian.

Tabel 3. Daftar modul pengujian *black box*

No	**Menu/ Fungsi**	**Butir Uji**
1	*Create*	*User* membuat *E-learning*
		Menampilkan pesan sukses
		Mengalokasikan ip untuk *container*
		Mengalokasikan *port* untuk *container*
		Menambah konfigurasi sites pada Nginx
		Menambah konfigurasi bind9
		Membuat konfigurasi Moodle
		Menyimpan data *container* pada basis data
2	*Start*	*User* memulai *container*
		Menampilkan pesan sukses
3	*Stop*	*User* memberhentikan *container*
		Menampilkan pesan sukses
4	*Delete*	*User* menghapus *container*
		Menampilkan pesan sukses
		Menghapus data pada basis data